VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikační rámec pro správu herního simulátoru Live for Speed
.NET Framework for Live for Speed Game

2012 Martin Janota

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Janota**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Aplikační rámec pro správu herního serveru simulátoru Live for Speed
.NET Framework for Live for Speed Game

Zásady pro vypracování:

Závodní simulátor Live for Speed se těší velké oblibě i v našich končinách. Hra nabízí dedikovaný server, který je ovládán textovými příkazy. Toto rešení je sice funkční, nicméně ne příliš uživatelský přívětivé a automatizovatelné.

Cílem diplomanta tedy bude nastudovat příkazy a vlastnosti dedikovaného serveru, navrhnout vhodný aplikační rámec pro jeho obsluhu a ten posléze i implementovat v technologii .NET.

Funkčnost navrženého řešení bude ověřena v podobě administrační aplikace, která bude schopna za běhu spravovat herní server.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Tomáš Kocyan**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení Studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě 3.5.2012

Poděkování

Touto cestou bych rád poděkoval svému vedoucímu, Ing. Tomáši Kocyánovi, za cenné rady a odbornou pomoc při realizaci této práce. Dále bych rád poděkoval svým blízkým za podporu během celého studia.

Abstrakt

Tato bakalářská práce se zabývá vytvořením knihovny tříd pro zpracování statistik získaných ze hry Live for Speed. Live for Speed je závodní hra, která je určena především pro více hráčů. Jejím hlavním úkolem je umožnit hráčům, pokud možno, co nejvěrněji simulovat řízení závodních aut. V této hře je možno jezdit se závodními auty na různých typech tratí. Hráči jezdí rychlá kola a zaznamenávají se jejich časy. Vzhledem k velkému počtu různých kombinací aut a tratí je, ale nutno tato data zachovávat, protože uživatel si nemůže tolik různých časů zapamatovat. K tomuto slouží hlavní server hry, který všechny tyto informace ukládá a poté je schopen tato data uživatelům, po zadání určitého dotazu, zprostředkovat. Cílem práce je tedy vytvořit knihovnu, která bude zprostředkovávat komunikaci mezi uživatelem a hlavním serverem.

Abstract

This bachelor work deal with creation of class library for processing statistics obtained from Live for Speed game. Live for Speed is racing game that is designed primarily for multiplayer. Its main task is to allow gamers as close as possible to simulate the driving of racing cars. In this game there is possible to drive racing cars on different type of tracks. Players run fast laps and their times are recorded. Due to large number of the different combinations of cars and tracks is needed to save these data because of user can't remember so many different times. For this is the main server of game which saves whole this information and then is able to mediate these data to users after entered of query. The aim of the work is therefore to create a library that will mediate communication between the user and the main server.

Klíčová slova

Live for Speed, XML, server, statistiky

Key words

Live for Speed, XML, server, statistics

Seznam použitých symbolu a zkratek

LFS	- Live for Speed
XML	- Extensible Markup Language - jazyk je určen především pro výměnu dat mezi aplikacemi
.NET Framework	- prostředí potřebné pro běh aplikací
C#	- programovací jazyk
PHP	- Personal Home Page – skriptovací programovací jazyk
MB	- megabyte – jednotka informace
TCP	- Transmission Control Protocol – protokol ze základní sady protokolů internetu
IP	- Internet Protocol – protokol používaný v počítačových sítích a Internetu
HTTP	- Hypertext Transfer Protocol - protokol určený pro výměnu hypertextových dokumentů ve formátu HTML

Obsah

1 Úvod 1

2 Aplikační rámec pro hru LFS 2

2.1 Hra Live for Speed 2

2.2 Komunikace mezi servery hry LFS 2

2.3 Úvodní rozhraní 3

2.4 Nastavení serveru za chodu 4

2.5 Spojení mezi uživatelským serverem a grafickým programem 5

2.6 Důvod změny náplně práce 5

3 Dotazy na hlavní server 6

3.1 Slovníček pojmů 6

3.2 LFS Free pubstats 6

3.3 Druhy dotazu 6

3.3.1 Povinné atributy každého dotazu 7

3.3.2 Nepovinné atributy každého dotazu 7

3.4 Identifikační klíč 8

3.5 Tratě 9

3.6 Auta 10

4 Komunikace s hlavním serverem 12

4.1 Možné odpovědi hlavního serveru 12

4.1.1 Odpověď ve tvaru XML souboru 12

4.2 Odpovídání serveru s časovými intervaly 13

4.3 Rozeznávání nepovinných atributů 13

5 Analýza 14

5.1 Datová analýza 14

5.1.1 Obsažené třídy 14

5.1.2 Třídní diagram modelu analýzy 14

5.3 Vstup od uživatele 16

5.4 Struktura a způsob ukládání dat 16

5.5 Rozlišení vstupního textu příp. XML souboru 16

5.6 Řešení problematiky načítání ... 16
6 Návrh ... 18
6.1 Požadavky na návrh ... 18
6.2 Třídní diagram modelu návrhu ... 18
6.3 Seskupení dotazů ... 19
6.4 Sjednocení dotazů ... 19
6.5 Využití třídní struktury ... 20
6.6 Strukturování souboru pro ukládání ... 20
6.6.1 Stahování text nebo XML ... 20
6.6.2 Struktury jednotlivých dotazů ... 21
6.7 Rozdělení dotazů na jednotlivé metody ... 23
6.8 Omezení zadávání dotazů a atributů ... 25
6.9 Možnost přímého načítaní ... 25
6.10 Postup zpracování přijatého dotazu ... 26
6.11 Aktualizace dat ... 27
6.12 Nastavení programu ... 27
7 Testovací aplikace ... 28
7.1 Požadavky na testovací aplikaci ... 28
7.2 Jednoduchý vzhled ... 28
7.3 Plná funkčnost ... 29
7.4 Ukázka praktického využití ... 31
8 Závěr ... 32

1 Úvod

Sledování statistik a to především takových, ve kterých je člověk sám aktérem hry, zajímalo lidstvo snad odjakživa. V dnešní době vyspělých technologií se tento vývoj nezměnil. Naopak se rozvíjí o možnost být hráčem počítačových her. Jednou z takových her je i hra Live for Speed, ve které má hráč možnost vybrat si auto a stát se automobilovým závodníkem. Snem každého jezdce je snaha o zdokonalování svých řidičských kvalit a zlepšování svých dosažených časů. Hra Live for Speed nabízí hráčům několik verzí tratí a modelů aut. Vzhledem k rozsáhlosti hry, co se týče možných kombinací tratí a aut, dosáhne hráč hry mnoha možných zajetých časů. Nicméně není v lidských možnostech si všechny své časy přesně pamatovat. Hra Live for Speed ovšem všechny tato data ukládá, aby lidé měli možnost porovnání časů nejen svých, ale i například mezi svými známými.

Tato práce se zabývá zpracováním těchto statistik a dat, které jsou přijaty z hlavního serveru hry Live for Speed, který si všechny tyto informace ukládá a dává k dispozici pro využití.

Hlavním účelem této práce je tato data přijmout, postupně zpracovat a využít užitečná data. Tento server může poskytnout mnoho užitečných dat. Vše záleží na druhu uživatelem zvoleného dotazu. Těchto dotazů existuje celkem 10 a umožňují uživateli zjistit v podstatě vše, co hráč během svého působení ve hře vykonal, např. kolik ujel kol, jestli je zrovna online, jaké časy zajel na jednotlivých tratích apod.

Kapitola 2 se věnuje popsání hry samotné a úvodního zadání s vysvětlením změny zadání. V kapitole 3 jsou popsány různé způsoby, jak je možné zadat dotaz na hlavní server pro získání dat, které uživatel chce získat. V další kapitole, čili kapitole 4, jsou popsány problémy, které vznikají při zadávání dotazů na hlavní server. V dalších kapitolách je popis, jak analýzy, tak návrhu řešení. Nakonec je popsána testovací aplikace, kterou bylo nutno vytvořit pro otestování funkčnosti.

2 Aplikační rámec pro hru LFS

Úvodní zadání této bakalářské práce, se týkalo zpracování jednoduchého grafického rozhraní pro hostující servery uživatelů hry Live for Speed(dále jen LFS). Jednoduché rozhraní pro tyto servery již existuje, ale je špatně ovladatelné s nedobrými ovládacími prvky.

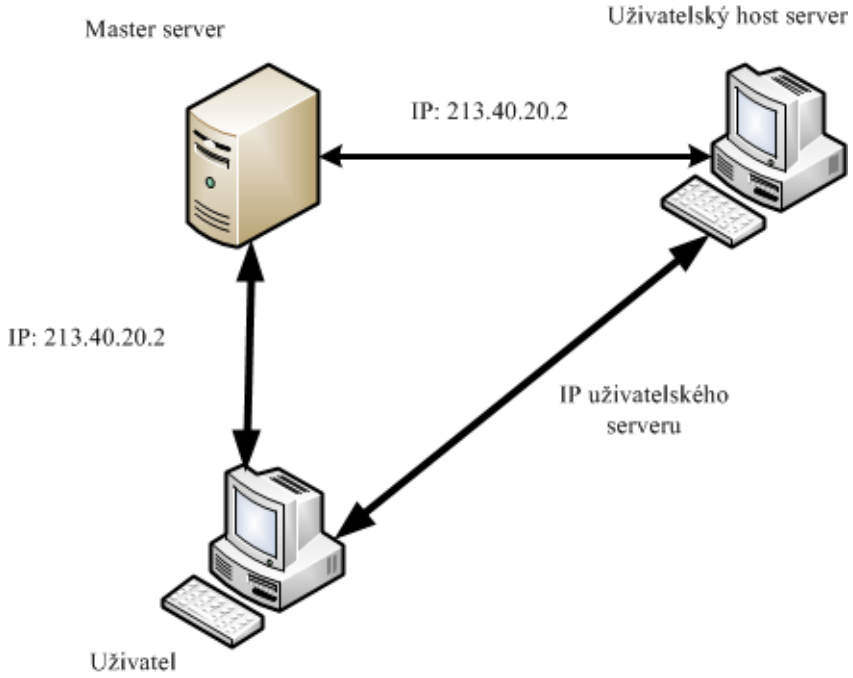
2.1 Hra Live for Speed

Live for Speed je hra, která se snaží co nejvěrněji simulovat závody vozidel. Jednou z největších předností LFS je jeho fyzikální model - podoba s chováním skutečného vozidla je opravdu velice blízká. Ovšem LFS nabízí i mnoho dalšího, a to online závody, do kterých je možno se přihlásit a soupeřit tak, se svými známými, o co nejlepší výsledky.
Jedná se o volně stažitelnou hru.

2.2 Komunikace mezi servery hry LFS

Ve hře LFS existují dva druhy serverů. Uživatelský server a hlavní server hry (viz obrázek 1).
Hlavní server (také nazýván Master server) hry LFS je její nejdůležitější součástí. Slouží k několika účelům. Tyto účely jsou následující:

1) Hostovat uživatelské servery, na kterých mohou uživatelé hrát.
   Na obrázku 1 lze vidět znázornění komunikace mezi hlavním serverem hry, hostujícími servery uživatelů a uživateli samotnými.



Obrázek 1 - schéma komunikace mezi servery a uživatelem

Jak lze vidět, tak uživatelské servery komunikují s hlavním serverem hry pomocí IP protokolu a to konkrétně na adrese `213.40.20.2`. Hlavní server poté vytváří seznam těchto serverů. Tento seznam je pak dostupný pro uživatele, kteří si spustí online hru a chtějí se připojit na některý ze serverů. Následně, když si uživatel vybere server, na kterém chce hrát, tak je přesměrován na IP adresu hostujícího serveru, kde již může hrát.

2) Zaznamenávat všechna data, která uživatelé „vyprodukují“.
3) Odpovídat na dotazy ohledně statistik uživatelů.

Druhý typ serveru je uživatelský server. Uživatelských serverů může být mnoho. Jsou to servery, které provozují na svých počítačích samotní hráči. Tyto servery jsou připojeny k hlavnímu serveru hry, se kterým po celou dobu existence komunikují.

2.3 Úvodní rozhraní

Na obrázku 2 je vidět úvodní rozhraní pro host server uživatelů, který je k dispozici na stránkách hry LFS.

Obrázek 2 - hlavní okno uživatelského serveru

K nastavení je poté možno použít jednoduchý konfigurační soubor, kde se nastavují všechny potřebné informace před startem herního serveru. Mezi tyto nastavení patří například jméno serveru, heslo nutné pro připojení na server, počet aut a jejich typy, trať, na které bude závod

probíhat a spousta jiných informací. Tyto informace, lze během chodu serveru ponechat, anebo je také za chodu měnit. Právě změna informací za chodu programu je jádrem problému.

2.4 Nastavení serveru za chodu

Program je možno za chodu programu nastavovat a měnit tak informace o serveru. Je možno provést například restart serveru, změnu trati a spoustu dalších, ovšem je nutno znát spoustu příkazů k tomu určených a tím se ovládání stává docela složitým.
Všechny příkazy jsou sice přehledně seřazeny v jednom textovém souboru, je jich však velké množství a uživatel si je všechny nemůže pamatovat.
Zadávání příkazů do běžícího serveru se provádí jednoduchými příkazy, které začínají lomítkem. Na obrázku 3 lze vidět zadání příkazu *reinit*, který způsobí celkový restart serveru. Na tomto obrázku je také možné spatřit odpověď serveru, což bylo restartování do 3 sekund.

Obrázek 3 - zadávání příkazu do serveru

Z obrázku je rovněž patrné, že server si ihned po restartu znovu načetl základní konfiguraci. Naprosto stejným způsobem, jako zadání příkazu *reinit*, se zadávají i všechny ostatní příkazy a ty se dělí na několik druhů:

- Jednoduché příkazy bez parametru – mezi tyto příkazy se řadí například *restart* nebo již výše zmíněný *reinit.*

- Příkazy s parametrem ve vstupním módu – parametr se zde zadává tak, že je přidána mezera za příkaz a za mezerou je napsán parametr. Tyto příkazy jsou například počet kol anebo nastavení počasí.
- Příkazy se vstupním parametrem, které je možno zadat kdykoliv – zadávání je totožné jako v předchozím případě. Mezi tyto příkazy se řadí nastavení maximálního počtu uživatelů, kteří se mohou na server připojit, rezervování míst pro administrátory a tak dále.
- Příkazy upravující závody v *Autocross* módu – zadávají se s parametrem a jsou to například nastavení počtu kol módu *Autocross*.
- Zákaz / Odpojení / Sledování – jsou to příkazy, které slouží administrátorům k ovládání hráčů na serveru. Mohou je vyhodit nebo jim přikázat pouze sledování atd. Zadávají se spolu s jedním parametrem a tím je jméno uživatele.
- Příkazy pro tresty – tyto příkazy jsou vkládány spolu se jménem uživatele. Slouží k trestům během závodu, například za porušení rychlosti v boxové uličce.
- Příkazy ovládající zobrazené zprávy – jsou to zprávy, které se zobrazují uprostřed hracího okna uživatelů. Mohou být poslány všem, kteří jsou připojeni anebo jen jednomu uživateli.

Dále existuje mnoho jiných příkazů, které je možno zadávat jen přímo ve hře.

2.5 Spojení mezi uživatelským serverem a grafickým programem

Spojení na uživatelský server je prováděno prostřednictvím služby *InSim*[1]. *InSim* zprostředkovává také rozesílání zpráv všem uživatelům, kteří jsou momentálně na uživatelském serveru připojeni. Tento přenos zpráv může být využit k mnoha věcem. Jsou to například spravování host serveru pomocí externího programu, získávání statistik ze serveru a podobně.
InSim služba využívá k připojení na uživatelský server spojení přes TCP protokol.

2.6 Důvod změny náplně práce

Důvod změny náplně práce je následující. Během vytváření původní aplikace, došlo k zásadnímu problému, který byl takový, že byla vydána nová verze uživatelského host serveru. V této verzi již nebylo možné naplno využívat všechny možnosti k řízení serveru. Omezení se týkalo docela velké části z možných příkazů, které umožňovaly ovládání uživatelského serveru. Tímto se zmenšily možnosti k řízení serveru a práce by byla tímto hodně omezena. Proto bylo nutné zajistit nové zadání. Nové zadání se tedy nebude týkat vytváření grafické aplikace pro komunikaci s uživatelským host serverem, nýbrž se bude týkat komunikace s hlavním serverem. Tato komunikace bude obsahovat zaslání dotazu na hlavní server hry a přijímání jeho odpovědí.

3 Dotazy na hlavní server

Dotaz směřující na hlavní server je ve formátu, který lze vidět v kapitole 3.3. Jedná se o klasický http dotaz. Po zadání dotazu, hlavní server hry pošle uživateli odpověď, která obsahuje v těle text. Tento text závisí na tvaru daného dotazu a vždy se vztahuje právě k tomu zadanému dotazu. Uživatel si může samotný dotaz jednoduše vyzkoušet tak, že ho ve správném tvaru zadá do jakéhokoliv internetového prohlížeče.

3.1 Slovníček pojmů

Ve hře LFS je k dispozici značný počet různých pojmů, které jsou velmi známé pro hráče. Mnoho z nich je samozřejmě také důležité znát, pokud chce uživatel pracovat s herními statistikami. Zde je výběr toho, co je důležité vědět pro práci se statistikami.

- *Drag* - závody na krátkou vzdálenost s pevným startem v přímém směru.
- *Hot lap* - měřené kolo, mód v LFS, při kterém jsou zaznamenávány závodníkova data.
- *Host* - uživatelský server, sloužící závodníkům k možnému ježdění kvalifikací či ostrých závodů.
- *PB* - "Personal Best" - osobní rekord za jedno kolo na trati.
- *WR* - "World Record" - světový rekord za jedno kolo na trati v *hotlap* módu.
- *Fuel* - palivo potřebné pro zajetí jednoho kola. Udávané v procentech nádrže.

3.2 LFS Free pubstats

Díky *LFS free pubstat*[2] lze přistupovat ke všem statistikám, které jsou k dispozici. Tyto statistiky si ukládá hlavní server hry, na kterém jsou v provozu všechny servery uživatelů. *LFS free pubstats* uživateli umožňuje aby, si zjistil všechny své statistiky, kterých ve hře dosáhl. Bohužel, je zde problém, že server neodpovídá na požadavky neustále, ale pouze jednou za pět sekund, aby nebyl přetížen opakujícími se požadavky. Existuje možnost využít *LFS premium pubstats*[2], který je ovšem placený. Platí se za každý požadavek zaslaný na server a za každý MB dat stažených ze serveru. Nejsou to nijak závratné částky, ovšem existují různá fóra a webové stránky, na kterých jsou tato data zobrazena. Uživatelé na nich zobrazují data nejen své, ale i svých spoluhráčů a je zbytečné platit za data, jež využívají i jiní uživatelé. Navíc je velká pravděpodobnost, že na těchto fórech, jsou data zobrazovány i s menším odstupem než je 5 sekund a tak by vznikaly chyby při zobrazování těchto dat.

3.3 Druhy dotazu

LFS pubstats nabízí 9 možných dotazů plus jeden pomocný. Každý dotaz vrací rozdílná data.
Tvar dotazu:
Každý dotaz, který je posílán na hlavní server, musí mít předem daný tvar. Začátek každého dotazu je vždy stejný a to takovýto:

`http://www.lfsworld.net/pubstat/get_stat2.php` za tímto vstupním odkazem následují specifika každého dotazu. Specifik je více a některé jsou volitelné a některé jsou povinné.

3.3.1 Povinné atributy každého dotazu

Každý dotaz má tři povinné atributy a těmito atributy jsou:

- Verze – udává, které číslo verze má být pro dotaz použito. Verze se liší například počtem atributů přijatým v odpovědi, nebo některé dotazy v nejnižších verzích úplně chybí. Nejnovější verze je 1.5. Knihovna pracuje s verzí 1.4.
- Identifikační klíč – možnost identifikování se, byla přidána ve verzi 1.2 a od verze 1.3 je povinná. Používání *pubstats* není možné bez identifikačního kódu. Identifikační klíč je blíže popsán v sekci 3.4.
- Akce – akce je poslední část povinných atributů, na kterou již přímo navazuje jeden z nepovinných atributů.

Dotaz vypadá po zadání těchto atributů následovně:
`http://www.lfsworld.net/pubstat/get_stat2.php?version=<Číslo_verze>&id k=<Identifikační_klíč>&action=…`

3.3.2 Nepovinné atributy každého dotazu

O tom, jaký dotaz je právě použit, rozhodují nepovinné atributy dotazu. Z již zmíněných 10 druhů musí být vybrán právě jeden. Takže neexistuje možnost dotazy kombinovat anebo nezadat ani jeden nepovinný atribut. Tyto atributy, se zadávají vždy za rovnítko u *action* a většinou vyžadují doplnění některých dalších údajů jako je např. jméno závodníka, trať atd. Některé z nich, lze doplnit i o další údaje, které slouží pro upřesnění dotazu. V této části se budou vyskytovat různé pojmy, jako *hot lap*, *world record* atd. Všechny tyto pojmy jsou popsány ve slovníčku pojmů v kapitole 3.1.
Tyto atributy tedy jsou:

- *Help* – zobrazí nápovědu, kde je vše potřebné popsáno
- *Hl* – tento dotaz, vrací vždy nejlepší čas v *hot lap* módu, kterého závodník dosáhl. Těchto časů je několik, vždy záleží na tom, s kolika auty a na kolika tratích závodník jezdil. Vyžaduje zadání jména závodníka.
- *Ch* – dotaz, který vypíše tabulku *hot lap* pro danou trať a auto. Vrátí tabulku, ve které jsou pro danou trať a auto seřazené nejlepší časy závodníků.
  Vyžaduje zadání trati a auta.
- *Wr* – zobrazí *world record* vždy pro každé auto a trať. Existuje možnost nezadání žádného atributu, díky které se zobrazí všechny *world record* hry LFS. Další možností je zadat pouze jeden z atributů, auto nebo trať, v tom případě, budou vypsány informace ohledně toho jednoho zadaného atributu. Poslední volbou, která se nabízí, je zadat oba atributu současně. V tomto případě se zobrazí data zohledňující oba současně zadané atributy.
- *Pb* – vrátí závodníkovy nejlepší časy na všech tratích a se všemi auty.

Vyžaduje zadání jména závodníka.

- *Fuel* – vypíše palivo potřebné pro ujetí jednoho kola daného závodníka.
  Vyžaduje zadání jména závodníka.
- *Pst* – pomocí tohoto dotazu, lze získat všechny užitečné informace o závodníkovi. Tyto informace jsou následující: Kolik závodník za celou dobu ujel metrů, počet zajetých kol, množství serverů, na které se závodník připojil, kolikrát se umístil na prvním, druhém a třetím místě, kolikrát dojel závod, počet zajetých kvalifikací, počet pole umístění, kolikrát se zúčastnil *drag* závodu, počet vítězství v *drag* závodu, ze které je země, jaký je zrovna jeho online status, a poslední známé auto a trať, které řídil a kde jel.
  Vyžaduje zadání jména závodníka.
- *Hosts* – zobrazí informace o všech serverech uživatelů, na kterých je momentálně možno závodit. Lze získat také informace o závodnících právě závodících na těchto serverech.
- *Teams* – vrátí informace o týmech, které závodí v LFS světě. Informace o tom, kolik má členů, jejich jména a jiné. Je možno zjistit informace o všech týmech v LFS nebo jen zadat jméno jednoho týmu a zjistit informace o něm.
- *Counters* – globální informace o autech a zajetých kolech s nimi nebo o tratích a zajetých kolech na nich.
  Vyžaduje zadání auta nebo trati a verze musí být vyšší než 1.4.

Zadání dalších pomocných atributů se provádí tak, že se za zkratku nepovinného atributu vloží symbol & a za něho se dopíše pomocný atribut a poté rovnítko s hodnotou pomocného atributu. V tomto případě existuje několik druhů pomocných atributů.
Těmito pomocnými atributy mohou být:

- *Racer* – za kterého se zadává jméno závodníka.
- *Track* – zde se zadává zkratka trati.
- *Car* – slouží pro zkratku auta.
- *Team* – pro vložení jména týmu.
- *Type* – tento pomocný atribut se nachází pouze u nepovinného atributu *counters* a slouží k určení, zda budou získány informace o autech nebo tratích. Úloha tohoto pomocného atributu je pouze k zadání *cars* anebo *tracks*.

Tímto jsou vyčerpány všechny možné dotazy, které lze v *pubstats* použít a tak výsledný dotaz může vypadat následovně:

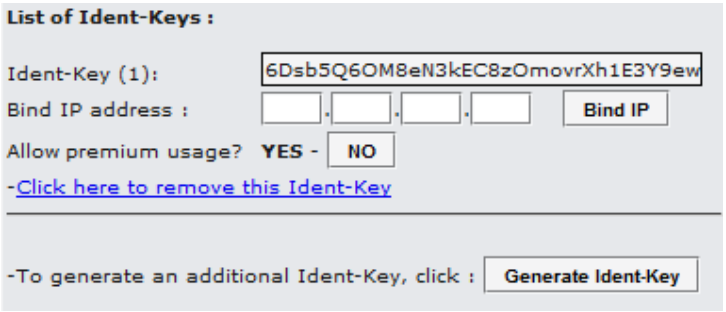
```
http://www.lfsworld.net/pubstat/get_stat2.php?version=<Číslo_verze>&id
k=<Identifikační_klíč>&action=hl&racer=<Jméno_závodníka>
```

3.4 Identifikační klíč

Identifikační klíč je možnost, jak identifikovat vlastní osobu při zadání dotazu na server. Byla přidána ve verzi 1.2, kde byla nepovinná, avšak od verze 1.3 je povinností každého, kdo chce získávat statistiky z *pubstat*, ať už *premium pubstat* nebo *free pubstat*, zadat tento klíč do každého dotazu, který posílá na server. Byl zaveden proto, aby si mohli tvůrci hry, udělat

obrázek o tom, jak jsou tyto statistiky využívány, neboť právě *pubstat* tvoří velkou část zátěže hlavního serveru hry, který kromě zvládání těchto dotazů musí ještě „obsluhovat“ herní servery uživatelů. Další důvod zavedení je také možnost vytvoření *premium pubstats*. Pro získání identifikačního klíče je nutná registrace na stránkách hry a vytvoření herního účtu.
Existují dvě možnosti jak získat identifikační klíč:

1) Tento klíč je možné vygenerovat na stránkách *LFS world settings*. To následujícím způsobem:
   Na stránkách `www.lfsworld.net`, po přihlášení uživatelským jménem a heslem, se objeví obrazovka, kde je možnost získávat informace o světě LFS. Na této obrazovce se po zvolení My *LFSW settings,* kde se objeví oknu a pod záložkou *Pubstat access* je možné vygenerovat vlastní identifikační klíč.
   Na obrázku 4 lze vidět vygenerování klíče, přičemž jeden klíč je už vygenerovaný a je možno přidat další klíč.



Obrázek 4 - generování uživatelského klíče

2) Druhá možnost je místo identifikačního klíče používat v dotazu uživatelské jméno a heslo. Zadávalo by se přímo do dotazu tak, že `>&idk=<Identifikační_klíč>` by se nahradilo tímto `&user=<username>&pass=<password>`. Ovšem operovat s heslem v dotazu není moc doporučeno, vzhledem možnosti zjištění hesla třetí osobou.

3.5 Tratě

V LFS existuje 7 různých fiktivních oblastí pro závodění. Každá z těchto oblastí má několik různých konfigurací a také je možné každou z těchto konfigurací, pokud je okruhového typu, jet v opačném směru. Celkem existuje 54 různých kombinací a tři z tohoto počtu jsou *Rallycross*. *Rallycross* je specifický v tom, že na něm nelze jezdit se všemi auty. Například s formulí je na něm v módu Hot lap zakázáno jezdit.
Tratě se zadávají do dotazu pomocí své číselné zkratky, která je jim přidělena na základě určitého schématu. Tato číselná zkratka je třímístná.
Zkratka se skládá z těchto tří částí. První část označuje z jaké oblasti je daná trať. Druhé číslo udává konfiguraci oblasti. Třetí číslo určuje, jestli trať bude v klasickém nebo obráceném stylu.

Seznam tratí
Zde je seznam všech oblastí a jejich možných konfigurací včetně jejich zkratky.

V prvním sloupci je uveden název oblasti, v druhém konfigurace a v posledním sloupci je zkratka tratě pro zadávání dotazu.

Blackwood	GP track	000
	Rallycross	010
South City	Classic	100
	Sprint 1	110
	Sprint 2	120
	City Long	130
	Town Course	140
	Chicane Route	150

Fern Bay	Club	200
	Green	210
	Gold	220
	Black	230
	Rallycross	240
	Rallycross Green	250

Autocross	Autocross	300
	Skid pad	310
	Drag strip	320
	8 Lane drag	330
Kyoto ring	Oval	400
	National	410
	GP Long	420
Westhill	International	500

Aston	Cadet	600
	Club	610
	National	620
	Historic	630
	Grand Prix	640
	Grand Touring	650
	North	660

Tabulka 1 – seznam tratí

Všechny výše uvedené tratě lze jezdit i v opačném směru. Existuje pouze jedna oblast, která je v tomto výjimkou. Tato oblast je *Autocross*. Tratě v oblasti *Autocross* totiž slouží hlavně pro jiné účely a to pro *drag*. Pokud je zvolena trať v opačné konfiguraci, tak pouze na konci její číselné zkratky je uvedena jednička místo nuly.

3.6 Auta

V LFS existuje 20 různých aut. Každé z těchto aut je určeno svým jménem a svou zkratkou. Auta nejsou ve většině kopií reálných aut, ale pouze jsou to auta vymyšlená tvůrci. To ovšem nemění nic na tom, že je možné na nich nastavit spoustu věcí jako u reálných aut. Zkratka auta je vždy třímístná a je odvozena ze jména auta. Na rozdíl od zkratky tratě, jsou zkratky aut složeny ze tří písmen.

Seznam aut

Zde jsou v tabulce uvedena všechna auta včetně jejich zkratek. Při zadávání dotazu směřujícího na hlavní server, je nutné zadat zkratku auta nikoliv jeho celé jméno.

V prvním sloupci je uveden název auta a v druhém sloupci je uvedena zkratka auta.

Název auta	**Zkratka auta**
XF GTI	XFG
XR GT	XRG
Formula BMW	FBM
XR GT Turbo	XRT
RB4 GT	RB4
FXO Turbo	FXO
LX4	LX4
LX6	LX6
MRT5	MRT
UF 1000	UF1
RaceAbout	RAC
FZ50	FZ5
XF GTR	XFR
UF GTR	UFR
Formula XR	FOX
Formula F8	FO8
BMW Sauber F1	BF1
FXO GTR	FXR
XR GTR	XRR
FZ50 GTR	FZR

Tabulka 2 – seznam aut

4 Komunikace s hlavním serverem

Jak již bylo výše popsáno, tak problém spočívá v tom, že jsou vytvořeny dotazy, které jsou posílány na hlavní server, a ten posílá zpět různé druhy odpovědí. Možné odpovědi hlavního serveru záleží na tom, jaký dotaz je mu poslán nebo na tom jestli je vytvořený dotaz správný.

4.1 Možné odpovědi hlavního serveru

Odpovědi hlavního serveru se mohou lišit. Uživatel například nemusí zadat správný identifikační klíč nebo například může zadat nesprávné jméno závodníka. Všechny tyto odpovědi mohou způsobit problém, neboť není nikde předem definována jedna chybná odpověď, díky které by se rozpoznal nesprávně zadaný dotaz.
Možné odpovědi:

- Správná odpověď – správná odpověď může být ve tvaru dlouhého textu, který začíná ve většině případů identifikačním číslem kola, případně ujetou vzdáleností nebo číselnou zkratkou trati. Další možnost správné odpovědi může být ve tvaru XML souboru. Odpověď ve tvaru XML souboru je více popsána v kapitole 4.1.1.
- `hl: no hotlaps found` – při zadání dotazu na získání statistik ohledně *hot lap* je možná tato odpověď. Značí špatně zadané jméno závodníka nebo zadané jméno neexistujícího závodníka.
- `hl: no racer` – odpověď při dotazu na *hot lap* a nezadání jména závodníka.
- `ch: invalid track` – při zadávání dotazu pro získání dat ohledně tabulky *hot lap*. Tato odpověď je zobrazeno při zadání špatné trati. Tato odpověď může být i ve tvaru, ve kterém je namísto *track* uveden *car*, a to když je zadáno špatné auto.
- `pb: no valid username` – při zadání špatného jména závodníka nebo při nezadání žádného jména. Objeví se při zadávání dotazu ohledně *personal best* závodníka. Stejný výpis hlavní server pošle, i když je dotázán na *fuel* závodníka, jen na začátku je uveden *fuel* namísto *pb*. Tento výpis, je také uveden při dotazu *pst*, opět je na začátku výpisu uvedeno *pst*.
- `No output` – tento výpis je zobrazen při jakékoliv chybě v dotazu, například nesprávně zadaném nepovinném atributu.
- `Invalid Ident-Key` – objeví se při zadání nesprávného identifikačního klíče
- `can't reload this page that quickly after another` – zobrazí se, pokud je dotaz načítán s malým odstupem a to menším než 5 sekund, během kterých hlavní server neodpovídá na dotazy jednoho uživatele s *Free pubstat*.

4.1.1 Odpověď ve tvaru XML souboru

Pro to, aby byla získána odpověď ve tvaru XML souboru, je potřeba na závěr dotazu přidat koncovku `&s=3`, díky které server pošle jako odpověď strukturovaný XML soubor.
Tento výpis se obzvláště hodí při zadávání dotazu na týmy nebo hostující servery, kdy server posílá místo klasického textového výpisu soubor PHP. PHP soubor je, ale nevhodný na zpracování v tomto případě, kdy jsou data zpracovávána v jedné knihovně.
XML soubor je strukturován poměrně jednoduše, ale ne moc přehledně, na další zpracování kvůli několika vnořeným prvkům.

4.2 Odpovídání serveru s časovými intervaly

Problém nastává také v tom, že server neodpovídá neustále, ale jak již bylo popsáno v kapitole 3.2 s určitým časovým odstupem. Tento problém výrazně ztěžuje práci se statistikami, neboť pokud je potřeba kontinuální načítání statistik. Pokud by bylo načítání statistik umístěno na nějakou webovou stránku, tak by nastal problém, že při přístupu dvou uživatelů v jeden okamžik by ten, který přišel o něco málo dříve, měl zobrazeny kompletní statistiky a druhý by měl znázorněn pouze výpis o chybovém hlášení.
Toto ovšem není přípustné a tak je nutné tento problém nějak vyřešit. Řešení toho problému spočívá v tom, že se aktuální přijatá data jednoduše uloží do ideálně strukturovaného souboru, aby bylo možné tato data následně jednoduše načíst a přehledně zpracovat.
Ovšem nastává zde další problém. Tím je takzvané stárnutí dat. Stárnutí dat způsobuje to, že pokud není dlouho zadán určitý dotaz, tak se data uložená v souboru stávají s přibývajícím časem stále méně aktuálními. To znamená, že pokud jsou následně tyto data načtena, tak uživatel dostává neaktuální informace.
Problematika řešení toho problému je dále popsána v následujících kapitolách.

4.3 Rozeznávání nepovinných atributů

Dotazy, které má uživatel na výběr, je zapotřebí nějak od sebe odlišit. Přestože má uživatel několik voleb, tak je nutné předem zjistit, jakou si právě vybral. Toto je nezbytné z důvodu toho, že každý ze zadávaných dotazů, je dále rozlišen pomocí nepovinných a pomocných atributů. Tak pro každý nepovinný atribut tedy musí být vytvořena zvláštní metoda, která bude zpracovávat pouze daný dotaz. Je pak pouze na uživateli, aby tyto metody využíval ve správnou chvíli a vybral si metodu, která slouží ke zpracování jeho vybraného dotazu.

5 Analýza

V této kapitole bude popsána datová analýza programu. Tato analýza bude obsahovat datovou analýzu, včetně třídního diagramu modelu analýzy a také bude naznačovat řešení načítání a ukládání dat.

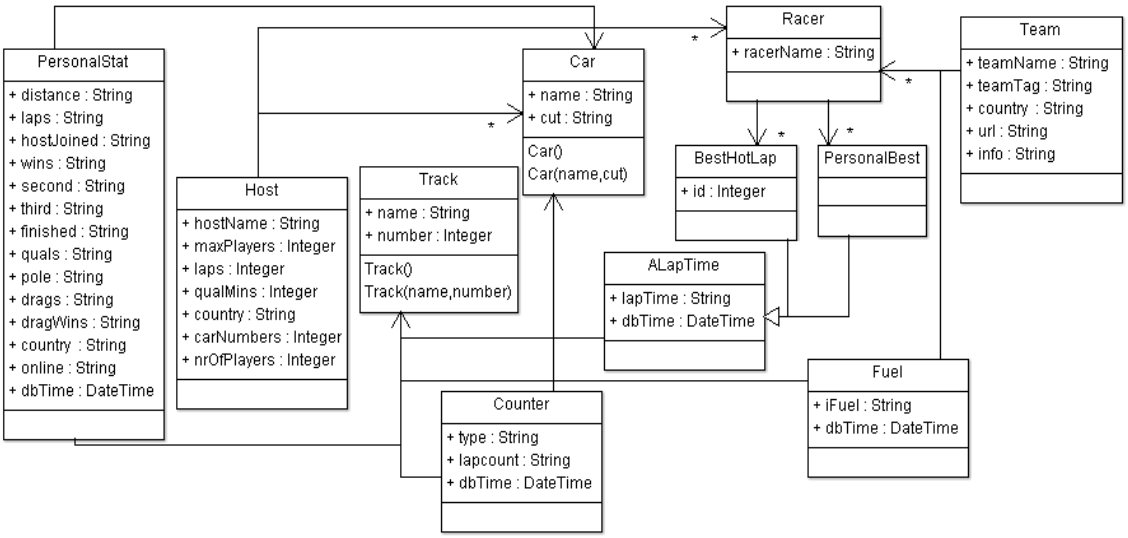
5.1 Datová analýza

Pro dobře navrhnutou třídní strukturu je nejprve vhodné pochopení toho, jaké třídy by měla tato struktura obsahovat. Poté tyto všechny prvky uvést do přehledného diagramu.

5.1.1 Obsažené třídy

Každý z dotazů, které si uživatel může zvolit má odlišnou vnitřní strukturu, tak je nutné pro každý tento dotaz udělat vlastní třídu. Dále je nezbytné všechny tratě a auta mít přehledně uložené ve vlastní třídě, ve které budou reprezentovány svým jménem a zkratkou. Zahrnuta je také třída *ALapTime*, která je abstraktní třídou.

5.1.2 Třídní diagram modelu analýzy



Obrázek 1 – třídní diagram modelu analýzy

Na obrázku 3 je možno vidět model analýzy vytvořený pomocí třídního diagramu.
Seznam tříd s uvedením toho co reprezentují:

- *Host* – třída reprezentující uživatelský host server. Jsou v ní uloženy všechny potřebné informace, jako je jméno serveru, na kolik kol se závod jede apod. Je v asociaci s třídami *Track, Car a Racer*, kde jsou informace o tom, na jaké trati se závod jede, s jakými auto je možno na serveru jezdit a kteří uživatelé jsou momentálně připojeni.

- *Counter* – třída reprezentující dotaz *counters*. Obsahuje tři atributy, kterými jsou, *type, lapcount a dbTime*. Tyto atributy charakterizují použitý typ. Tedy zdali se jedná o dotaz směřující na získání informací ohledně tratí nebo aut, počet kol ujetých na určité trati nebo s určitým autem a také čas uložení do databáze. Jak je možno odvodit z předchozího popisu, tak tato třída je v asociaci s třídami *Track* a *Car*. Přitom používá vždy právě jednu z těchto tříd po zadání dotazu.
- *PersonalStat* – třída, jenž reprezentuje statistiky uživatele získané po dotazu *personal stat*. Obsahuje mnoho atributů, které charakterizují uživatelovy osobní statistiky. Mezi tyto atributy patří například atribut, podle kterého lze určit, zda je daný závodník on-line. Třída je v asociaci s třídami *Track*, *Car* a *Racer*. Tyto třídy obsahují informace o autu a trati, které závodník naposledy využil a v třídě *Racer* je samozřejmě závodníkovo jméno, které se získá po zadaném dotazu.
- *Track* – tato třída slouží k reprezentaci tratí. K tomuto účelu má i dva atributy, které charakterizují jméno a zkratku trati. Jsou s ní v asociaci všechny třídy, které zastupují dotazy, s jednou výjimkou, kterou tvoří třída *Teams*.
- *Car* – třída, která je podobná třídě *Track*. Ovšem třída *Car* reprezentuje auta, nikoliv tratě. Obsahuje taktéž dva atributy, které slouží k určení jména a zkratky auta. Asociace této třídy jsou totožné jako u třídy *Track*.
- *Team* – třída reprezentující týmy. Obsahuje atributy, které charakterizují tým. Těmito atributy například jsou jméno týmu nebo země odkud tým pochází. Tato třída je v asociaci pouze s třídou *Racer*. Třída *Racer* bude uchovávat informace o tom, kteří závodníci jsou momentálně členy týmu.
- *Racer* – třída, jež má za úkol reprezentovat závodníka. Pro tento účel má jeden atribut, který jednoznačně určí, o kterého závodníka se jedná. Tímto atributem je závodníkovo jméno.
- *BestHotLap* – třída, která zastupuje více než jen jeden dotaz. Reprezentuje celkově tři dotazy. Těmito dotazy jsou *hot lap*, *hot lap chart* a *world record*. Tyto dotazy pracují vždy se stejnými atributy, a proto je možné pracovat s nimi v jedné sloučené třídě. Třída obsahuje dva zděděné atributy z třídy *ALapTime* a také atribut *id*, podle kterého se určuje *ID hot lap* anebo *world record*. Třída je zděděnou třídou třídy *ALapTime*.
- *PersonalBest* – třída reprezentující závodníkovy nejlepší časy získané pomocí dotazu *personal best*. Obsahuje pouze zděděné atributy třídy *ALapTime*, tedy jak z toho vyplývá je to zděděná třída.
- *ALapTime* – je to nadřazená třída pro třídy *BestHotLap* a *PersonalBest*. Obsahuje dva atributy sloužící k charakterizování času kola a času uložení do databáze. Je v asociaci s třídami *Track, Car*, a *Racer*. Jedná se o abstraktní třídu.
- *Fuel* – třída reprezentující dotaz *fuel*. Tedy třída, jež charakterizuje palivo potřebné pro ujetí okruhu na určité trati. Má dva atributy a to právě informace o palivu a čas uložení do databáze. Je v asociaci s třídami *Track, Car* a *Racer*.

5.3 Vstup od uživatele

Podle počtu možných vstupních dotazů existuje tedy devět možných vstupů do programu. Je třeba je jednoznačně odlišit a určit uživateli, kdy může jaký použít.
Proto je třeba navrhnout 9 různých metod zpracování vstupního dotazu. Tyto metody budou muset být komplexní, neboť je třeba pracovat s možností různého výběru volitelných atributů. Některé z dotazů mají předem dán počet volitelných atributů, kde si uživatel vybírá pouze ze seznamu tratí nebo aut, ovšem existují i dotazy jako *wr(world record)*, kde je možné vstupní dotaz zadat, jak ve tvaru bez volitelných atributů, tak ve tvaru s jedním nebo oběma volitelnými atributy.
Proto je třeba navrhnout komplexní metody, které budou reagovat na všechny možné situace, a to i na takové, že se v programu vyskytne chyba.

5.4 Struktura a způsob ukládání dat

Data, která jsou přijata z hlavního serveru, je nutno uchovávat pro pozdější využití, v případě, kdy hlavní server neodpovídá správnými informacemi.
Ukládání dat je možné několika způsoby. Pokud by, ale uložená data měla sloužit nadále i k pozdějšímu načítání, pak je nutno tato data ukládat strukturovaně.
Jeden z nejlepších možných strukturovaných způsobů ukládání dat, je pomocí XML souboru.
Vzhledem k tomu, že server ve většině případů posílá data ve formě textu, jenž se zpracuje na potřebné části podle struktury odpovědi, které se pak ukládají do instancí tříd zvoleného dotazu, tak je vhodné si tato data přehledně uložit do XML souboru.
Struktura ukládání se různí v závislosti od použitého dotazu. Nelze totiž použít jeden univerzální soubor nebo jednu univerzální strukturu, ve které by mohla být uložena všechna data. Proto je nutné, pro každý dotaz navrhnout vlastní strukturu. Některé dotazy mají sice jednu vlastní abstraktní třídu, jak je patrné na třídním diagramu v kapitole 5.1.2, ovšem ani některé z nich nemají jednotnou strukturu vzhledem k jinému vstupnímu textu.
Struktury jednotlivých XML souborů jsou popsány v kapitole 6.6.2.

5.5 Rozlišení vstupního textu příp. XML souboru

Každý z dotazů má svůj vlastní tvar textu, který vrací hlavní server. Tento tvar je předem daný. I v tomto případě je tedy vhodné rozlišení zpracovávání na 9 metod stejně, jako když je rozlišován vstup od uživatele. Vstupní text je přímo vázán na vstup od uživatele. Takže stačí rozlišit vstup od uživatele a s tím je pak už ulehčená práce se vstupním textem nebo XML souborem.

5.6 Řešení problematiky načítání

Neboť hlavní server nevrací správnou odpověď neustále, ale s časovými rozestupy, je vhodné také určit způsob, jak tyto správné odpovědi ošetřit a „schovávat“ pro případy, kdy je to vhodné. Je tedy třeba nutné ošetření toho, kdy bude dotaz veden až na server a kdy to není bezpodmínečně nutné. Protože se spolu se staženými daty bude v XML souboru uchovávat i čas, kdy byla tato data stažena, tak je možnost tento čas kontrolovat a zjišťovat, jak aktuální

data jsou. Pokud nejsou tato data příliš neaktuální, tak je možnost data rovnou načíst, aniž by byla velká pravděpodobnost toho, že jsou tato data během krátké chvíle změněna a načetly by se tím pádem data neaktuální.
Nicméně u každého dotazu se liší čas, během kterého se stávají data neaktuálními. Například u dotazu *hot lap*, je daleko méně pravděpodobné, že závodník zajel během krátké chvíle nějaký lepší čas. Ovšem například u dotazu *counters*, se mění data každou chvíli, neboť s každou chvílí přibývají zajetá kola, ať už na nějaké trati nebo s jakýmkoliv autem.
Proto je třeba tento čas zvolit nějak rozvážlivě a to pro všechny tyto dotazy.

6 Návrh

Po datové analýze je potřeba udělat také návrh, ve kterém již budou specifikovány použité třídy a taktéž se tento návrh bude komplexně zabývat načítáním a ukládáním dat z programátorského hlediska.

6.1 Požadavky na návrh

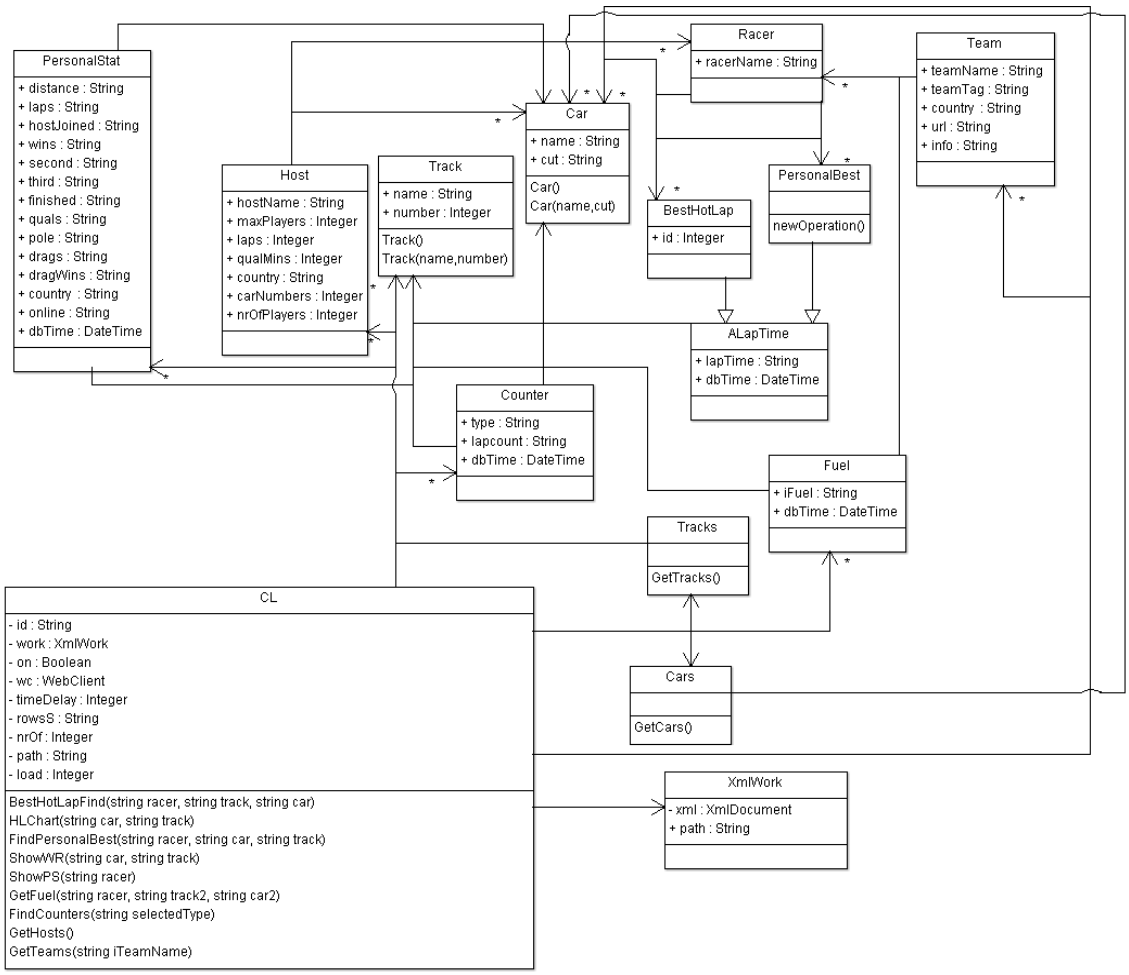
Je nutné stanovit určité minimální požadavky, které musí návrh splňovat.

- Seskupit všechny dotazy pod jednu knihovnu.
- Sjednocení dotazů se stejným základem.
- Využít navrhovanou třídní strukturu v knihovně.
- Vytvořit ideálně strukturované soubory pro ukládání dat.
- Nechat uživateli možnost zadávat jen volitelné atributy a možnost výběru dotazu.
- Rozdělit zadávání dotazů na jednotlivé metody.
- Realizovat možnost přímého načítání.
- Jednoduché nastavení knihovny.

6.2 Třídní diagram modelu návrhu

Na následujícím schématu (Obrázek 6), lze vidět třídní diagram vytvořený dle modelu návrhu. Oproti modelu analýzy přibyly 4 třídy.
Následuje seznam nových tříd s vysvětlením toho, co reprezentují.

- *Tracks* – zastupuje vytváření seznamu aut. Je v asociaci s třídou *Track*, jíž využívá pro získání dat ohledně jedné trati.
- *Cars* – jedná se o obdobný případ jako třída *Tracks*, ovšem slouží k vytvoření seznamu aut. Je v asociaci s třídou *Car*.
- *XmlWork* – třída reprezentující metody, které slouží pro práci s XML souborem. Má obsaženo hodně metod, které nejsou vzhledem k přehlednosti diagramu uvedeny. Obsahuje také dvě proměnné. Jedna *xml*, která reprezentuje *XmlDocument*, a druhá *path*, která slouží k určení cesty pro ukládání XML souboru. Je v asociaci s třídou *CL*.
- *CL* – *CL* je hlavní třída celé knihovny, jako jediná obsahuje metody, které má uživatel možnost využít. Obsahuje proměnné sloužící k nastavení celé knihovny. Celá třída se skládá z mnoha metod. Hlavních devět metod, které využívá uživatel je zobrazeno v třídním diagramu. Ostatní nejsou pro přehlednost uvedeny. Je v asociaci se všemi třídami, jež jsou v návrhu, kromě třídy *ALapTime*.



Obrázek 2 - třídní diagram modelu návrhu

6.3 Seskupení dotazů

Přestože jsou dotazy rozdílné, je třeba je seskupit tak, aby uživatel mohl využít pouze jednu knihovnu pro všechny dotazy. Bylo by zbytečné mít každý dotaz rozdělený ve své zvláštní knihovně. Tohoto lze jednoduše docílit tím, že se zpracování každého dotazu sjednotí do jedné knihovny tříd. Tím bude poté možno se všemi dotazy pracovat jako s jedním celkem a bude poté možnost využití dále.

6.4 Sjednocení dotazů

Ačkoliv se všechny dotazy zadávají rozdílně, tak některé z nich poskytují stejná data. Bylo by proto zbytečné, úplně rozdělit jejich zpracování a hlavně ukládání. Je proto třeba tyto dotazy rozlišit a využít toho, že se z nich získávají stejná data.
Toto se týká právě dvou ze všech dotazů. Konktrétně dotazů hot lap a hot lap chart. Kdy dotaz *hot lap*, vrací informace ohledně *hot lap* jednoho závodníka. Oproti tomu dotaz *hot lap chart*, vrací tabulku nejlepších závodníků a jejich *hot lap* na určité trati. To znamená, že pokud by se uživatel dotazoval na *hot lap* závodníka X, poté na tabulku *hot lap* na jedné trati, ve které by byl

také závodník X obsažen a data se ukládaly do dvou různých souborů, tak by vznikala redundantní data. Bylo by proto nevhodné tato data ukládat do dvou souborů.
Dle toho je ideální využít jeden ukládací prostor pro oba tyto dotazy.

6.5 Využití třídní struktury

Aby byla knihovna co nejvíce optimalizována pro jednoduchou práci s ní, tak je nutno využít navrhovanou třídní strukturu. A to takovým způsobem, že uživatel bude na výstupu dostávat vždy instanci dotazu, který chtěl využít. Ovšem není možné, aby na výstupu byla pouze instance dotazu. Pokud je to dotaz směřující také na informace o jiných třídách například o tratích, tak je nutné, aby na výstupu byly také instance i jiných tříd. Proto je nutné v programu využít globální instance i těchto tříd. Těmito třídami jsou závodník, trať a auto resp. *Racer, Track* a *Car*. Takže uživatel poté bude moci využít nejen instancí tříd, na které byl veden dotaz, ale také instancí tříd, které se k dotazu vážou. Je pak jen na uživateli, jak tyto informace využije. Protože ovšem nelze využít jen jednu instanci třídy pro získání například celé tabulky *hot lap*, tak jen nutno si tyto instance nějak hromadně shromažďovat. K tomu účelu se dá skvěle využít nástroj *List v .NET Framework*. Jednotlivé instance se budou postupně zařazovat do seznamu a je možné je poté jednoduše využít. Ke každému seznamu bude také vázán vždy i seznam, který se bude skládat z instancí pomocných atributů. Tak bude možné jednotlivé informace snadno propojit mezi sebou.

6.6 Strukturování souboru pro ukládání

Jak již bylo uvedeno, vstupní data je nutno ukládat. Vzhledem k tomu, že je nutno využít strukturované ukládání dat, tak je ideální XML soubor. Ale protože existuje devět různých dotazů a skoro každý z nich má různou strukturu, je nutno vytvořit více různých struktur, neboť žádná univerzální struktura není možná vytvořit. Jedna struktura bude stejná, pro dotazy hot lap a hot lap chart. Ukládání do XML souboru nebude třeba v případech dotazů *hosts* a *teams*, protože odpovědi jsou stahovány ze serveru už v podobě XML souboru.

6.6.1 Stahování text nebo XML

Jak je patrné z předchozí kapitoly, tak je možnost stahování odpovědí z hlavního serveru přímo v XML souboru. Nabízí se otázka, proč toho nevyužít i u ostatních dotazů a vyvarovat se tak zpracovávání textu a ukládání do souboru. Odpověď je jednoduchá a souvisí s tím, že server neodpovídá neustále a také s tím, že jsou třeba ideálně strukturované soubory.
Vzhledem k tomu, že hlavní server neodpovídá neustále, ale s určitým časovým rozestupem, tak je vhodné, aby se data ukládaly zároveň s tím, kdy byly přijaty. To by nebylo umožněno v případě stáhnutí XML souboru přímo ze serveru. Také by nebylo možné využít odložené načítání a dotazy by byly na hlavní server kladeny neustále, bez možnosti ušetřit možnost dotazu v případě, kdy to není nutné, jak je to popsáno v kapitole 5.6. Dále se s XML souborem stahují i data, která nejsou v knihovně dále k využití, proto není nezbytně nutné tato data uchovávat. Každý dotaz by také měl svůj XML soubor a to i případě toho, že by se lišil jeden volitelný atribut a to by neodvratně vedlo k tomu, že by vznikla spousta XML souborů a nešlo by se v tomto počtu souborů dobře orientovat. Další důvod je také ten, že stáhnuté XML

soubory nejsou vhodně strukturovány. Stahování XML souborů v případě dotazů *hosts* a *teams* je, ale nezbytně nutné využít. Hlavní server totiž v těchto případech nevrací jako odpověď text, ale posílá soubor s příponou PHP. Soubor PHP je, ale pro zpracování takovýchto dat, nevhodný, proto se v daných případech využívá stahování dat ve formě XML souboru.

6.6.2 Struktury jednotlivých dotazů

Nyní už k samotným popisům struktur XML souborů pro jednotlivé dotazy.

1) **Dotazy *hl* a *ch***
   U těchto dotazů je struktura poměrně jednoduše odvoditelná. Kdy je dán závodník a je potřeba z přijatého textu odvodit jednotlivé časy kol pro určitou trať a určité auto. Z tohoto jednoznačně vyplývá, že ideální strukturu lze vytvořit uložením závodníkova jména a k tomu poté po množinou *HotLaps* přiřazovat už konkrétní *HotLap* s danými atributy.
   Zde je možno vidět celou strukturu:

```
<Racers>
  <Racer name="racerName">
    <HotLaps>
      <HotLap>
        <Track>zkratka_trati</Track>
        <Car>zkratka_auta</Car>
        <Time>čas_kola</Time>
        <DbTime>čas_uložení_do_DB</DbTime>
        <ID_HL>id_HotLap</ID_HL>
      </HotLap>
    </HotLaps>
  </Racer>
</Racers>
```

Kód 1 - popis ukládání dat do XML pro dotazy *Hot Lap* a *Hot Lap Chart*

2) **Dotaz *wr***
   Také vzhledem k vlastnostem tohoto dotazu, je poměrně jednoduché odvodit strukturu. Dle toho, že dotaz *world record* nepřijímá žádné volitelné atributy, je o to jednodušší ukládání a poté i pozdější načítání dat. Dotaz *world record* vrací text, ve kterém je obsaženo jméno závodníka, čas kola, trať a auto, proto je nasnadě všechna tato důležitá data uchovat pro pozdější použití. Struktura je složená tak, že pod každým *WR* jsou uchovány data o něm. Toto je stejné pro všechny možné kombinace tratí a aut.
   Zde je struktura XML souboru pro dotaz *wr*:

```
<WRS>
  <WR>
    <Track>zkratka_trati</Track>
    <Car>zkratka_auta</Car>
    <Racer>jméno_závodníka</Racer>
    <Time>čas_kola</Time>
    <DbTime>čas_uložení_do_DB</DbTime>
    <Id_WR>id_WorldRecord</Id_WR>
  </WR>
</WRS>
```

Kód 2 - popis ukládání dat do XML pro dotaz *World Record*

3) **Dotaz pb**

Dotaz *personal best* má podobnou strukturu, jako již výše zmíněné dotazy *hl* a *ch*. Tento dotaz se zadává stejně jako dotaz *hl*, tedy tak, že se jako volitelný atribut zadává pouze jméno závodníka. Proto i struktura XML souboru je velice podobná.
Zde je struktura XML souboru pro dotaz *pb*:

```
<Racers>
  <Racer name="racerName">
    <PB>
      <Track>zkratka_trati</Track>
      <Car>zkratka_auta</Car>
      <Time>čas_kola</Time>
      <DbTime>čas_uložení_do_DB</DbTime>
    </PB>
  </Racer>
</Racers>
```

Kód 3 - popis ukládání dat do XML pro dotaz *Personal Best*

4) **Dotaz *pst***

Sktruktura ukládání pro pro *personal stat* závodníka je dána tím, že každý závodník má předem dané jedny své statistiky, takže není nutné využívat více podelementů, které by více specifikovaly například to, na jaké trati ujel určitou vzdálenost. Ve skutečnosti je v XML souboru *pst* dotazu více atributů, ovšem pro názornost, jsou zde uvedeny pouze dva, což je vzhledem k množství přehlednější. Kompletní seznam atributů je uveden v třídním diagramu modelu analýzy.
Zde je možno vidět celou strukturu:

```
<Racers>
  <Racer name="racerName">
    <Distance>ujetá_vzdálenost</Distance>
    <Laps>ujetá_kola</Laps>
  </Racer>
</Racers>
```

Kód 4 - popis ukládání dat do XML pro dotaz *Personal Stat*

5) **Dotaz *fuel***

Dotaz *fuel* má opět poměrně jednoduchou strukturu, podobnou například na strukturu *hl*. Opět je zde XML soubor rozdělen na závodníky, kteří poté mají pod množinou *fuels* rozepsány všechny své záznamy, ve kterých je uvedeno množství paliva vždy na určité trati s určitým autem.
Zde je vidět celá struktura:

```
<Racers>
  <Racer name="racerName">
    <Fuels>
      <Track>zkratka_trati</Track>
      <Car>zkratka_auta</Car>
      <Fuel>čas_kola</Fuel>
      <DbTime>čas_uložení_do_DB</DbTime>
    </Fuels>
  </Racer>
</Racers>
```

Kód 5 - popis ukládání dat do XML pro dotaz *Fuel*

6) **Dotaz *counters***

Pravděpodobně nejsložitější dotaz na vytvoření struktury. Musí se zjišťovat, zdali jde o dotaz zjišťující informace ohledně tratí anebo aut. Proto také v základním (*root*) elementu jsou dvě možnosti dalšího pokračování a to v závislosti na použitém volitelném atributu v dotazu. Dále se XML soubor vyvíjí stejně, pouze jsou změněny názvy XML elementů. V následující struktuře, lze vidět možné podoby tvaru XML souboru. Pokud je v názvu XML elementu uvedeno lomítko, tak to znamená, že může být zvolena jen jedna možnost z těchto dvou. Navazující element ovšem musí být pojmenován v souvislosti s předchozím, což znamená, že není přípustné, aby se podelement jmenoval *iCars*, když hlavní element bude mít název *Tracks*.

Zde je možno vidět strukturu:

```
<Type>
  <Tracks/Cars>
    <iTracks/iCars>
      <Track/Car>zkratka_trati/auta</Track/Car>
      <Lapcount>počet_ujetých_kol</Lapcount>
      <DbTime>čas_uložení_do_DB</DbTime>
    </iTracks/iCars>
  </Tracks/Cars>
</Type>
```

Kód 6 - popis ukládání dat do XML pro dotaz *Counters*

6.7 Rozdělení dotazů na jednotlivé metody

Ačkoliv je nutno metody sjednotit pod jednu knihovnu, tak je třeba, aby byly zpracovávány odděleně. To nejen z důvodu jednoduššího zpracovávání, ale také pro přehlednější zadávání dat od uživatele. Také proto je nutné vytvořit jednu vstupní metodu pro každý dotaz, která bude vždy přijímat na vstupu volitelné atributy a na výstupu bude instance nebo seznam instancí daného dotazu. Je třeba, proto jasně určit, které metody budou jaké atributy přijímat a s jakými instancemi bude metoda zpracovávající určitý dotaz pracovat. Vzhledem k přijatým atributům se také dá odvodit, které instance jiných tříd budou také použity.
Přehled hlaviček jednotlivých metod a instance, s kterými pracují.

1) `public BestHotLap BestHotLapFind(string racer, string track, string car)`

   Jak lze jednoduše odvodit s názvu metody, tedy *BestHotLapFind*, jedná se o hlavičku metody hledající *hot lap*. Jako vstupní atributy tato metoda přijímá jméno závodníka, název tratě a název auta. Dle toho je také možné určit s jakými instancemi, jakých tříd bude tato metoda pracovat. Bude tedy pracovat s jednou instancí třídy *BestHotLap*, kde bude ukládat čas závodníka, *id hot lap* a také čas položení dotazu, který se také zapíše do XML souboru pro pozdější využití. Dále bude také pracovat s instancemi tříd *Racer, Track* a *Car*, kde vždy uloží vybrané jméno, trať či auto.
2) `public List<BestHotLap> HLChart(string car, string track)`

   Metoda zpracovávající dotaz *ch*, neboli *hot lap chart*, je již jiná než prvně zmíněná metoda. Pracuje se stejnými instancemi, ovšem nevyužívá vždy jen jednu instanci,

nýbrž používá kolekci, konkrétně kolekci *List,* instancí. Jako vstupní atributy, jsou zde použity název trati a název auta. Tento dotaz tedy pracuje s instancemi tříd *BestHotLap, Track, Car* a přestože není uvedena v hlavičce žádná spojitost, tak také pracuje s instancemi třídy *Racer*. Třída *Racer* je využita proto, že k *hot lap chart* je vždy nutno uvést i jméno závodníka, který daný čas dosáhl. Z každé třídy je vždy použita kolekce *List*. Data spolu související jsou vždy ukládána na stejnou pozici v kolekci každé instance, tzn. data uvedena například na třetí pozici kolekce závodníků, se vztahují k datům uvedeným na třetí pozici kolekce tratí.

3) `public List<PersonalBest> FindPersonalBest(string racer, string car, string track)`
   Hlavička metody *FindPersonalBest* prakticky odpovídá hlavičce metody v bodě 1. Avšak tato metoda pracuje opět s kolekcemi *List* instancí, neboť na rozdíl od první metody zpracovává informace o všech kombinacích tratí a aut, kde má závodník zajet svůj nejlepší čas. Z toho také vyplývá, s jakými instancemi bude pracovat. To s instancemi tříd *Racer, Track, Car* a samozřejmě s instancí třídy *PersonalBest.*
4) `public List<BestHotLap> ShowWR(string car, string track)`
   Další podobný případ je metoda zpracovávající dotaz *world record*, tedy *ShowWR*. Opět je zde nutnost ukládat kolekce *List* tříd *Racer, Track, Car* a tentokráte *BestHotLap*. Vstupní atributy mohou být zadány anebo nechány ve stavu *null*, dle přání uživatele a také dle toho, jestli uživatel chce vědět *world record* všech kombinací nebo jen pro určité auto anebo trať.
5) `public PersonalStat ShowPS(string racer)`
   Metoda zpracovávající dotaz na osobní statistiky závodníka používá jen jednu instanci třídy *PersonalStat*, protože lze vždy získat informace jen o jednom závodníkovi během jednoho dotazu. Dále metoda využívá instancí tříd *Racer, Track* a *Car*, kdy se do instancí tříd *Track* a *Car* ukládají data o naposledy použitém autu a trati na kterém závodník naposledy jel.
6) `public List<Fuel> GetFuel(string racer, string track2, string car2)`
   Zpracování dotazu ohledně paliva je podobné. Znovu jsou použity kolekce *List* čtyř tříd a to *Racer, Car, Track* a *Fuel*.
7) `public List<Counter> FindCounters(string selectedType)`
   Metoda *FindCounters* je určena pro zpracování ujetých kol s určitým autem nebo na dané trati. Vstupním atributem je pouze typ, který bude zpracováván. Také dle určeného typu bude zvoleno, s kterými instancemi se bude pracovat. Tato metoda tedy pracuje s instancemi tříd *Counters* a také *Track* nebo *Car* v závislosti na zvoleném typu. Vždy se také pracuje s kolekcemi *List* daných tříd.
8) `public List<Host> GetHosts()`
   GetHosts je metoda zpracovávající XML soubor s informace ohledně serverů uživatelů. Tyto informace obsahují také data ohledně toho, jaké auta jsou momentálně na serveru, na jaké trati se jede, kolik závodníku je na serveru a samozřejmě také jejich seznam. Tím, je také určeno to, že metoda pracuje s instancemi tříd *Host, Racer, Track* a *Car*. Přitom jsou opět použity kolekce *List*. *List* instancí třídy *Racer* je řazen tak, že jsou vždy za sebou uvedena jména závodníků podle jejich počtu na serveru, za nimi jsou uvedena jména závodníku

dalšího serveru a tak dále. Vždy, je počet jmen závodníku na serveru uveden podle toho, kolik jich je uvedeno v počtu závodníků na serveru.

9) `public List<Team> GetTeams(string iTeamName)`
Hlavička metody *GetTeams* je poměrně stejná jako hlavička metody předcházející. Jen je uveden jeden vstupní atribut, který slouží k vyhledávání jména určitého týmu. Je ovšem možné uvést tento atribut i ve stavu *null* a tím nechat možnost vyhledávání všech týmů. Pracuje s instancemi tříd *Team* a také *Racer*, vzhledem k tomu, že u každého týmu je také uveden i seznam závodníků. Opět se pracuje s instancemi tak, že jsou ukládány do kolekce *List*.

6.8 Omezení zadávání dotazů a atributů

Přestože uživatel bude mít volnou možnost výběru dotazu, tak je nutné, aby byla omezena možnost zadání nesprávného tvaru dotazu. Proto je nutné, dát uživateli pouze takové možnosti, aby se této chyby snadno vyvaroval. To bude docíleno tím, že se uživateli zpřístupní jen určité funkce programu. To takové, ve kterých bude pouze zadávat volitelné atributy.
Uživatel tedy bude mít zpřístupněné jen hlavní metody knihovny s tím, že při jejich volání, bude mít v hlavičce uvedena pouze přesná jména volitelných atributů nebo v případě dotazu *counters* bude uveden typ. Dle toho bude zcela eliminována chyba nesprávného zadání dotazu a bude se muset řešit jen možnost toho, že uživatel zadal nesprávné hodnoty na místa volitelných atributů. Samotný dotaz pak bude vykonán již přímo v knihovně, kde již uživatel nebude mít přístup. Je tedy pouze na uživateli jestli bude zadávat správné hodnoty, a tím zajistí bezproblémový chod svého programu.

6.9 Možnost přímého načítaní

Možnost přímého načítání dotazu, je z hlediska uživatele zajímavé vzhledem k tomu, že u některých dotazů není třeba neustálé aktualizování přijatých dat. Proto si uživatel může nastavit čas, během kterého se nebudou data načítat přímo z hlavního serveru, ale z XML souboru. K tomuto účelu je nutné s přijatými daty ukládat také čas, kdy byly uloženy. Proto je také nutné mít v každé třídě atribut *dbTime*, jenž je určen k ukládání tohoto času.
V programu je na to pak určena proměnná, kterou si uživatel může nastavovat. Ovšem při zadávání času musí být uživatel opatrný a dávat si pozor, u kterých atributů je možno nastavit dlouhý čas a také naopak, u kterých atributů není vhodné dlouhý čas nastavovat. Kupříkladu *world record* se mění minimálně a čas tedy může být nastaven i na 60 minut a není pravděpodobné, že by se během té doby údaje příliš měnily. Naopak dotaz *counters*, kde se počítají ujetá kola, se mění každou chvíli, a proto není vhodné nastavovat ho na příliš dlouhou dobu. Ovšem vše záleží na rozhodnutí uživatele a na tom, zda chce mít data aktuální nebo zda mu nevadí, že data nebudou příliš aktuální.
Tuto možnost je zbytečné využívat, když je program využíván minimálně a dotazy nejsou tím pádem zadávány příliš často. Jen když bude program využíván opravdu často, tak najde tato funkce využití a uživatel by ji měl ocenit.
V programu je tato funkce automaticky aplikována při každém zadání dotazu od uživatele, kdy se vždy zjistí uživatelem nastavený čas. Uživatel může zadávat čas v minutách, tento čas se poté

přičte k času uloženému v daném XML souboru, který je načten v závislosti na dotazu, a zjišťuje se tím, zdali je přičtený čas menší než čas, ve kterém byl dotaz zadán.
Zde je uvedena jednoduchá nerovnice.

$$t_x + t_u > t_a$$

tx – čas uložený v XML souboru
tu – čas zadaný uživatelem
ta – aktuální čas při zadání dotazu
Pokud je tedy levá strana nerovnice větší než ta druhá, tak se načtou data z XML souboru. Pokud je tomu naopak, tak je poslán dotaz na hlavní server. Všechny časy jsou zadávány v minutách.

6.10 Postup zpracování přijatého dotazu

Jakmile uživatel zadá jakýkoliv ze sedmi dotazů, vyjma dotazy *hosts* a *teams*, tak postup zpracovávání tohoto dotazu je následující:

1) Metoda zpracovávající dotaz nejprve zjistí, zdali je možné využít možnosti načíst uložená data z XML souboru. Nejdříve je tedy testován čas uložení do XML souboru s časem, jaký si zvolil uživatel pro možnost přímého načtení dat, což je podrobně popsáno v kapitole 6.9. Pokud jsou daná data, které chtěl uživatel zobrazit aktuální, tak jsou přímo načtena z XML souboru. Dále se tato data zapíší do svých příslušných instancí a metoda je skončena. Pokud data nejsou aktuální anebo nejsou příslušná data uložena, tak metoda pokračuje dále bodem 2.
2) Metoda dále pokračuje tak, že zjistí, jestli vstupní text získaný z hlavního serveru začíná na „can“, což značí, že dotaz nebyl položen poprvé během uplynulých pěti sekund, viz kapitola 4.1. Zde se načte nastavení z XML souboru, a pokud si uživatel zvolil možnost počkat, tak metoda 5 sekund počká a poté se znova spustí, aby bylo zajištěno to, že server odpoví správně. Jestli tato možnost není zvolena, tak se metoda pokusí získat data z XML souboru. Pokud jsou požadovaná data uložena tak jsou načtena. V případě, že data uložená nejsou, tak uživatel obdrží zprávu o tom, že požadovaná data není momentálně možnost získat.
3) Jestliže je vstupní text v jiném jakémkoliv jiném formátu, než je popsáno v bodě 2, tak metoda pokračuje tímto bodem. V tomto bodě metoda zkusí analyzovat vstupní text (*parse text*) a pokud je zjištěno, že vstupní text není možno správně analyzovat a rozebrat na části, které by bylo možno správně zapisovat do instancí, tak je zachycena výjimka a uživatel dostane na výstup zprávu, která se bude rovnat odpovědi hlavního serveru, tzn. v případě špatného zadání jména uživatele, při dotazu *hot lap,* se uživateli zobrazí `hl: no hotlaps found`, viz kapitola 4.1. Pokud je ovšem text správně zpracován, tak je poslán do XML souboru, kde se dále zpracovává a ukládá. Délka ukládání závisí na druhu dotazu a na množství dat, které dotaz obsahuje.

Délka zpracovávání každého dotazu je přibližně stejná. Metoda vždy načítá data z XML souboru anebo naopak do něho data ukládá, takže uživatel by neměl pocítit rozdíl mezi různým zpracováním dotazu. Menší prodlevy mohou nastat jen v případě zpracovávání dotazu, jenž je

popsán v bodě tři, kdy se data, pokud již byly uloženy, aktualizují (viz kapitola 6.11) a mazání starších dat tak může způsobit menší prodlení. Menší zpoždění také může nastat při možnosti, kdy je dotaz poprvé spuštěn a XML soubor se musí vytvořit.

6.11 Aktualizace dat

Nezbytně nutnou součástí bezproblémové, a hlavně efektivní práce knihovny, je také aktualizace dat, které jsou již uloženy v XML souboru. Data, která jsou zapsána do XML souboru je nutno aktualizovat a neuchovávat tak stejná, ovšem méně aktuální data. Proto se vždy s každým příchozím dotazem ověří, zdali již nejsou tyto informace jednou uloženy. Pokud ano, tak se starší informace nahradí novými aktuálnějšími.

6.12 Nastavení programu

Uživatel by měl mít možnost nastavit si knihovnu dle jeho uvážení. Vzhledem k tomu, knihovna tříd nelze univerzálně nastavit pomocí *app config*, tak je třeba k nastavení využít jiných prostředků.
Nastavení lze tedy dělat pomocí souboru, ve kterém budou všechny tyto možnosti předdefinovány, a uživatel tedy bude zadávat pouze jednoduchá data k nastavení. K tomu účelu lze jednoduše využít opět XML soubor. V něm budou již předepsány možnosti, které bude mít uživatel k nastavení.
Uživatel si tedy může nastavit počet řádků výpisu *hot lap chart*. Při této možnosti uživatel nastaví počet řádků výpisu a tím určí, kolik řádků bude mít tabulka výpisu.
Dále si může nastavit čas, který určí čas přímého načítaní, což je podrobně popsáno v kapitole 6.9.
Poté má uživatel možnost nastavit si identifikační klíč. Toto je nesmírně důležité, aby uživatel správně nastavil svůj identifikační klíč. Pokud zadá špatný klíč, tak knihovna nebude fungovat, protože se do dotazu musí zadat správný identifikační klíč.
Další možností pro nastavení, je také cesta pro ukládání XML souboru. Tímto si uživatel může nastavit, kde chce, aby se mu ukládaly XML soubory s uloženými daty. Jestliže žádnou cestu nenastaví, tak se budou soubory ukládat na stejné místo odkud je spuštěna aplikace.
Následující způsob jak nastavit práci knihovny je určení, zdali se bude čekat při zjištění toho, že server neodpovídá kvůli časové prodlevě čekat na správnou odpověď, viz 6.10 bod 2.
K nastavení musí uživatel nejenže zadat správná data do XML souboru, ale také toto nastavení načíst, pokud ho změní během chodu programu. Nastavení se vždy načte s každým spuštěním programu, ale uživatel má také možnost si nastavení změnit během chodu programu, aniž by musel program restartovat a znovu spuštěním nastavení načíst. Aktualizování nastavení je možno provést jednoduchým zavoláním metody *UpdateSettings*, kterou je možno kdykoliv zavolat a tím si znovu načíst nastavení z XML souboru.

7 Testovací aplikace

K ověření funkčnosti programu je nutno vytvořit aplikaci, která bude testovat, zdali dotazy fungují správně. Je nutno zjistit možnosti různých zadávání dotazů a tím eliminovat možné chyby při zadávání dotazů. Tato aplikace musí být navrhnuta tak, aby lehce navazovala na knihovnu tříd, ve které jsou všechny metody zpracovávající dotazy zabaleny. Proto je jako testovací aplikace použita aplikace *Windows Forms*, kde lze vše otestovat.

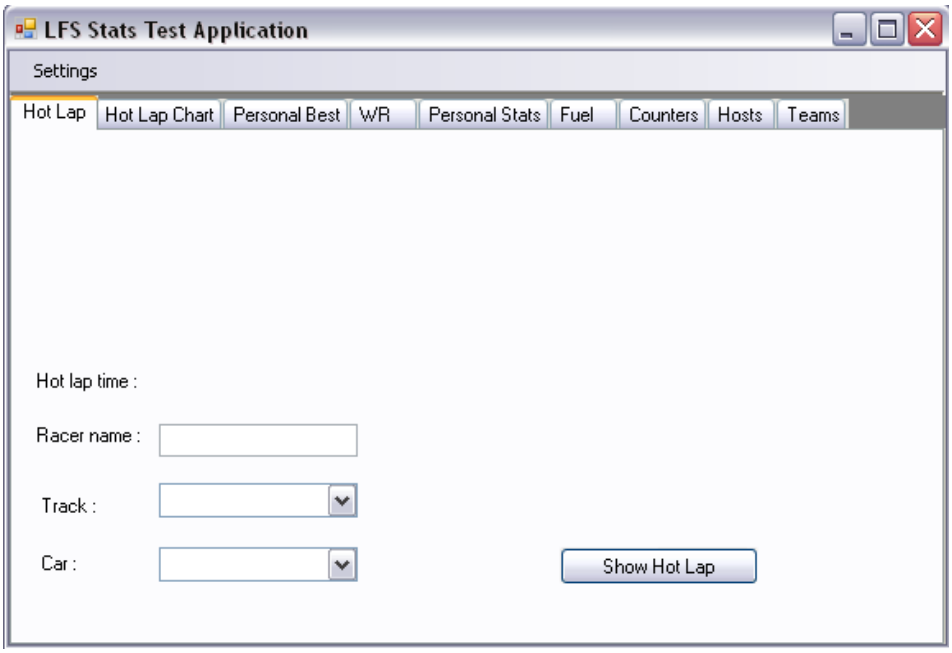
7.1 Požadavky na testovací aplikaci

Pro testovací aplikaci je třeba nastavit určité minimální požadavky, které musí splňovat.

- Jednoduchý vzhled
- Plná funkčnost

7.2 Jednoduchý vzhled

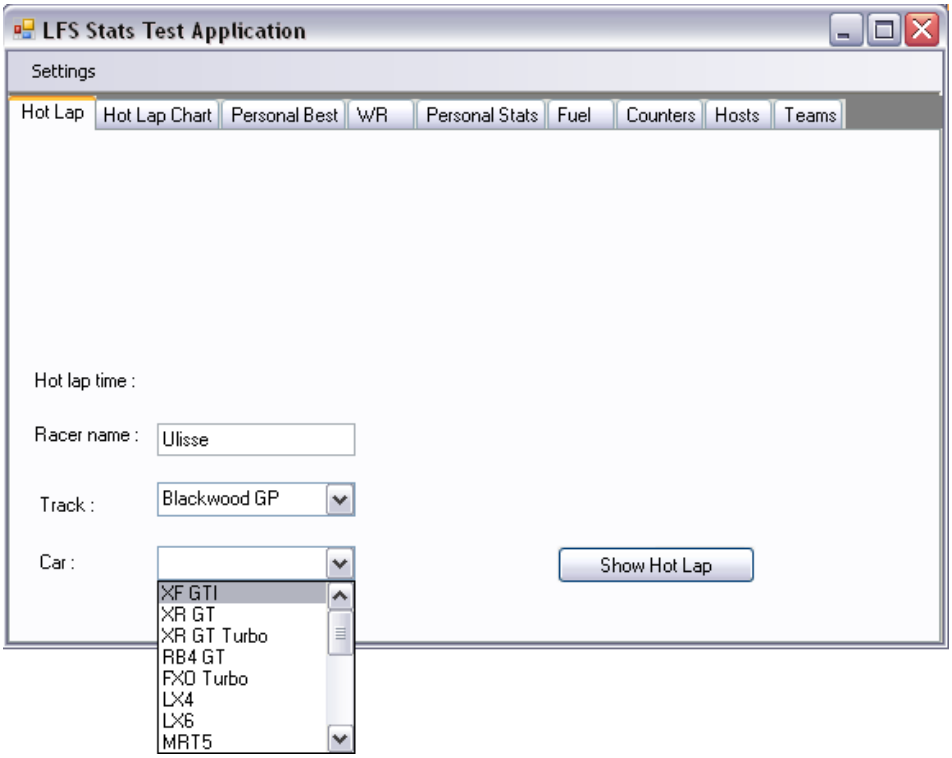
Vzhledem k tomu, že se jedná pouze o testovací aplikaci, tak není nutno, aby byl vzhled celé aplikace nějak zvláště upravován. To ovšem neznamená, že by vzhled mohl být nepřehledný. Na obrázku 7 lze vidět celý vzhled testovací aplikace.



Obrázek 3 – úvodní okno testovací aplikace

Na horní straně je možno vidět *Settings*, kde se může nastavit jednoduché nastavení programu a to konkrétně *ID*, počet řádků výpisu a také čas přímého načítání. Dále je pod tímto nastavením vidět devět panelů, které jsou určeny každý pro jeden dotaz. Na těchto panelech jsou vždy možnosti, jak zadávat volitelné atributy podle možností dotazu. Možnosti zadávání atributů jsou realizovány několika způsoby. Pro zadávání tratí a aut je vždy použit *ComboBox*, neboli seznam všech tratí a aut. Zadání závodníka se provádí jednoduše napsáním jeho jména do ovládacího

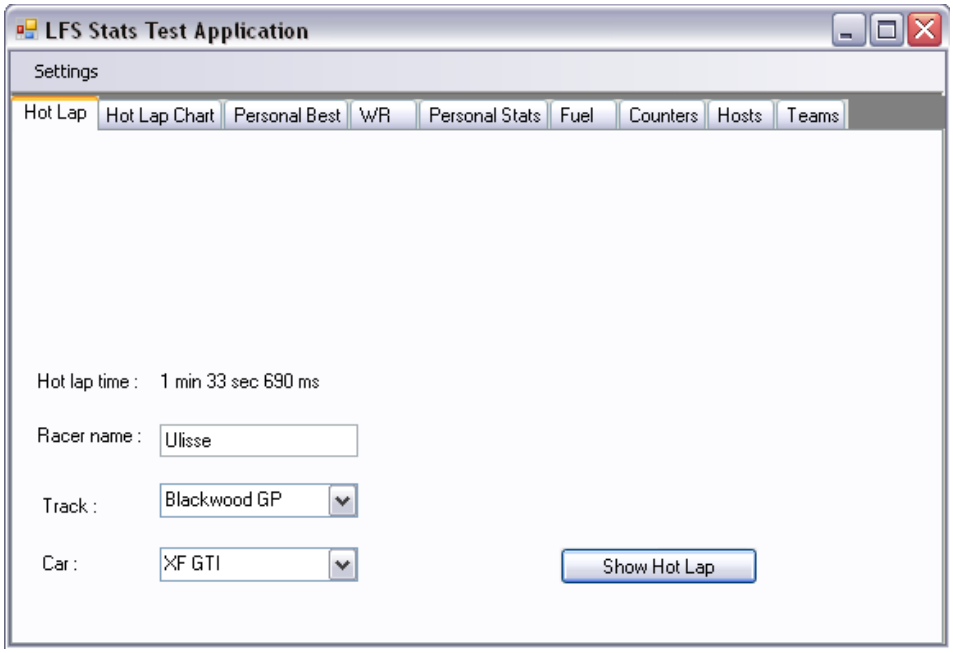
prvku *TextBox,* vždy u popisu *Racer name*, totéž řešení je použito na panelu *Teams* se zadáváním jména týmu. Na panelu *Counters* jsou pro zadávání typu použity dva ovládací prvky typu *RadioButton* pro přepínání zobrazení aut a tratí.
Obrázek 8 ukazuje zadávání atributů u dotazu *hot lap*.



Obrázek 4 – zadávání atributů v testovací aplikaci

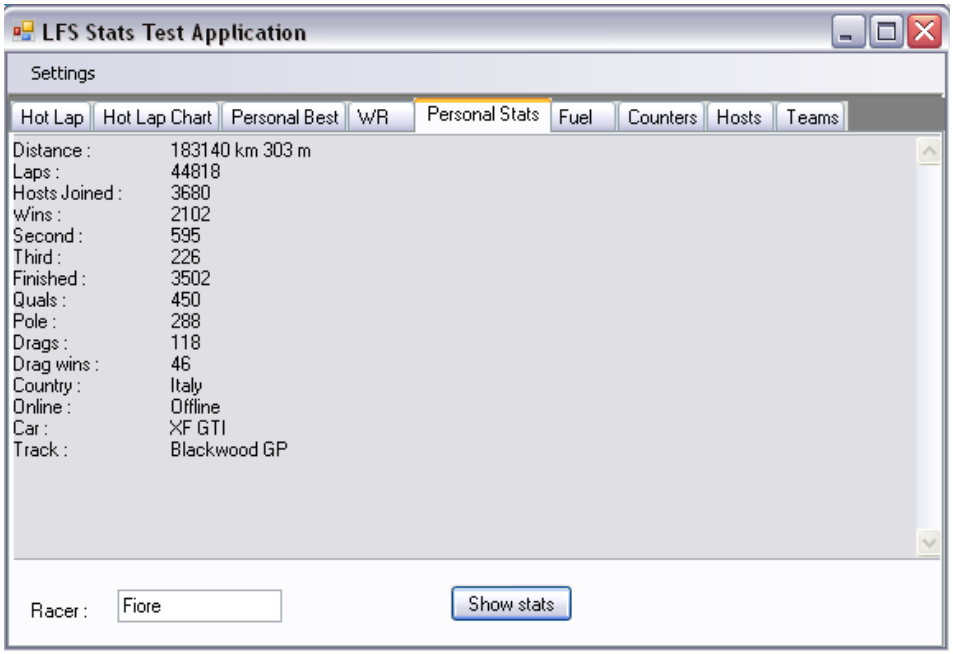
7.3 Plná funkčnost

K tomu, aby byla možnost zjistit, zda jsou metody napsány správně, je třeba také testovací aplikaci vytvořit tak, aby byly všechny funkce plně zprovozněny. K tomuto účelu je zapotřebí výsledky vypisovat na panely testovací aplikace, protože jen tak, lze snadno ověřit správnou funkčnost. Tomuto účelu je v programu vyhrazen velký prostor. Vždy je vyhrazený nějaký prostor k vypsání těchto dat, ať už to je jeden *Label* nebo celý *TextBox,* do kterého se vypisují vždy celé tabulky pro snadné zobrazení konečných výsledků.
Na obrázku 9 je zobrazeno plné zadání atributů do dotazu a také je u popisu *Hot lap time* zobrazen čas pro zvoleného závodníka, trať a auto. Čas daného kola je pak zobrazen v štítku (*Label*).



Obrázek 5 – výpis času kola

Ovšem ne vždy je vhodné vypisování pouze času. K vypisování většího množství dat, je tedy použit *TextBox*, který může být víceřádkový a tím umožnit přehlednější výpis. Tento výpis lze snadno vidět na Obrázku 10, který slouží pro výpis dotazu *personal stat*. Získaná data, jsou přehledně sepsána tak, aby bylo možno, se snadno orientovat v textu, a zjistit tak rychle potřebná data.

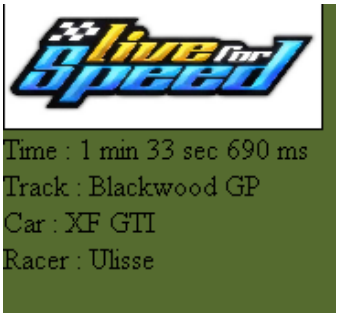


Obrázek 6 – zobrazení výpisu *personal stat*

Opět lze vidět jednoduché zadání atributu závodníkova jména a i zobrazení všech důležitých informací, které nám tento dotaz poskytuje.

7.4 Ukázka praktického využití

Jen testovací aplikace nestačí pro dostatečné otestování funkčnosti. Proto je vhodné vytvořit i aplikaci, která bude prakticky ukazovat využití knihovny tříd. K tomuto účelu je vhodná webová aplikace, která bude simulovat funkčnost knihovny na reálném fóru nebo na webové stránce. K ukázce funkčnosti je využita pouze jedna metoda pro zjišťování času kola *hot lap* určitého závodníka.
Na obrázku 11 lze vidět již zobrazenou grafickou stránku této webové aplikace.



Obrázek 7 - webová aplikace

V horní části obrázku jde vidět vložený obrázek, dle kterého jde jednoznačně poznat, o jakou hru se jedná. Dále je vypsán čas, který je zjištěn po zadání dotazu. Dále lze vidět trať, auto a jméno závodníka. Tyto poslední tři atributy jsou zadávány uživatelem do metody, kterou spouští z knihovny. Jak jednoduše je využita tato jedna metoda, tak stejně pracují i ostatní z knihovny.

8 Závěr

Cílem práce bylo vytvořit jednoduchou knihovnu tříd, která bude sloužit pro zpracovávání statistik stažených pomocí *WebClienta* z hlavního serveru hry Live For Speed. Program byl napsán v jazyce *C#*, ve kterém je již většina ostatních aplikací k této hře napsána. Aplikace bude velice vhodná pro spoustu uživatelů, kteří již tyto statistiky využívají, ale vzhledem k tomu, že nebyla dosud napsána žádná podobná aplikace, která by umožňovala práci se všemi dotazy najednou, tak byla práce s těmito statistikami komplikovanou záležitostí.
Zobrazování těchto statistik má již mnoho hráčů hry na svých webových stránkách či na různých fórech, ovšem jsou nejednotné anebo rozdělené. Díky této aplikaci již bude možné všechny tyto statistiky sjednotit a s využitím knihovny tříd tyto informace jednoduše zobrazovat.
Práce byla i pro mne velice přínosná, protože jsem si vyzkoušel, jak vypadá návrh celé takové knihovny tříd pěkně od začátku. Nejdříve jsem si navrhnul rozložení tříd pro jeden dotaz, to konkrétně pro dotaz *hot lap*, a na něm jsem pak dále testoval veškeré možnosti zadávání dotazu. Od úvodního dotazu se následně odvíjelo prakticky vše. Z úvodního návrhu jsem čerpal i nadále při tvorbě dalších dotazů, které pak rozšířily knihovnu tříd tak, aby obsahovala všechny dotazy. Nezbytnou součástí tvorby této knihovny tříd, byla i testovací aplikace, kterou sem vyvíjel postupně s knihovnou tříd vzhledem k nutnosti testovat funkčnost metod. Výsledkem byly funkční, jak knihovna tříd, tak testovací aplikace pro všechny dotazy.
Rozšíření knihovny tříd je možné poměrně jednoduchým způsobem. Mnou vytvořená knihovna tříd totiž zpracovává vstupní text z hlavního serveru tak, že filtruje informace a získává jen určitá data a to většinou o času kola a podobných důležitých věcech. Ovšem, je možnost tato data nefiltrovat a zpracovat všechny vstupní informace, které obsahují například informace o časech na jednotlivých úsecích tratí a podobně.
Využití tato aplikace určitě najde, jak je již výše zmíněno, na fórech, jako možné zobrazení statistik. Dále je možné využít testovací aplikaci. Testovací aplikace by mohla být jednoduše využita po upravení vzhledu všemi hráči hry, které zajímají jejich dosažené výsledky.

Seznam použité literatury

[1] Hráči Live for Speed, LFS Manual[online], dostupný z: <http://en.lfsmanual.net/wiki/Main_Page>

[2] Live for Speed Pubstats S2 v1.5 (19 Feb 2011)[online], dostupný z: <http://www.lfsworld.net/pubstat/get_stat2.php>

[3] Morgan Skinner, Christian Nagel, Ollie Cornes, Jay Glynn, Karli Watson, K.Scott Allen, Burton Harvey, Simon Robinson, Zach Greenvoss, C# Programujeme profesionálně, 1.vydání, COMPUTER PRESS 2003, 1130 stran, ISBN 80-251-0065-5

[4] Vondrák, Ivo. Úvod do softwarového inženýrství[online], dostupný z: http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

Seznam příloh

Obsah CD

Složka LFSstats

- kompletní solution včetně knihovny tříd a testovací aplikace

Složka LFS_CL

- samostatná knihovna tříd

Složka LFSstats

- testovací aplikace

Složka LFSWeb

- webová aplikace pro praktickou ukázku